# ES核心概念和原理

## 1、什么是搜索：百度、垂直搜索（站内搜索）

搜索：通过一个**关键词**或一段描述，得到你想要的（相关度高）结果。

## 2、如何实现搜索功能?

关系型数据库：性能差、不可靠、结果不准确（相关度低）

### 2.1例子

想要查询 “小米NFC智能手机”

|  |  |  |
| --- | --- | --- |
| id | brandName | describe |
| 1 | 小米手机 | 手机中的战斗机 |
| 2 | 小米NFC手机 | 小米手机,支持全功能NFC,手机中的战斗机 |
| 3 | NFC手机 |  |
| 4 | 小米耳机 |  |
| 5 | 红米耳机 |  |
| ...... |  |  |
| ...... |  |  |
| 10000000 | Xxx机器人 | 扫地好帮手 |
| ...... |  |  |

如果是sql查询，select \* from product where brandName like “%小米NFC智能手机%”，这种查询不走索引，会进行全表扫描，时间复杂度O(n) ，如果有1000w条数据就要有1000w次IO。再牛B的数据库也顶不住。

而且brandName字段数据也没有“小米NFC智能手机”导致搜不出来。

## 倒排索引、Lucene和全文检索

### 3.1引入倒排索引

倒排索引是基于document做的，这里为了方便理解用表做。

倒排索引是Lucene封装的一个功能，分词的第一步就是把词拆开，把上面的商品的brandName进行分词：

小米手机 ----------- 小米，手机

小米NFC手机 ----------- 小米，NFC，手机

NFC手机 ----------- NFC ，手机

小米耳机 ---------- 小米，耳机

红米耳机 ---------- 红米，耳机

这一行类似es的document list

|  |  |  |
| --- | --- | --- |
| 是否匹配到 | 索引 | 出现id |
| √ | 小米 | 1,2,4 |
| √ | 手机 | 1,2,3 |
| √ | NFC | 2，3 |
|  | 耳机 | 4，5 |
|  | 红米 | 5 |

|  |  |  |  |
| --- | --- | --- | --- |
| id | brandName | describe | 在匹配到结果中索引中出现次数  （匹配相关度） |
| 1 | 小米手机 | 手机中的战斗机 | 2 |
| 2 | 小米NFC手机 | 小米手机,支持全功能NFC,手机中的战斗机 | 3 |
| 3 | NFC手机 |  | 2 |
| 4 | 小米耳机 |  | 1 |
| 5 | 红米耳机 |  | 0 |
| ...... |  |  |  |
| ...... |  |  |  |
| 10000000 | Xxx机器人 | 扫地好帮手 |  |
| ...... |  |  |  |

### 3.2倒排索引数据结构

#### 3.2.2 ES数据结构

1. 包含这个关键词的document list
2. 关键词在每个doc中出现的次数 TF term frequency （词频）
3. 关键词在整个索引中出现的次数 IDF inverse doc frequency
4. 关键词在当前doc中出现的次数
5. 每个doc的长度，越长相关度越低
6. 包含这个关键词的所有doc的平均长度

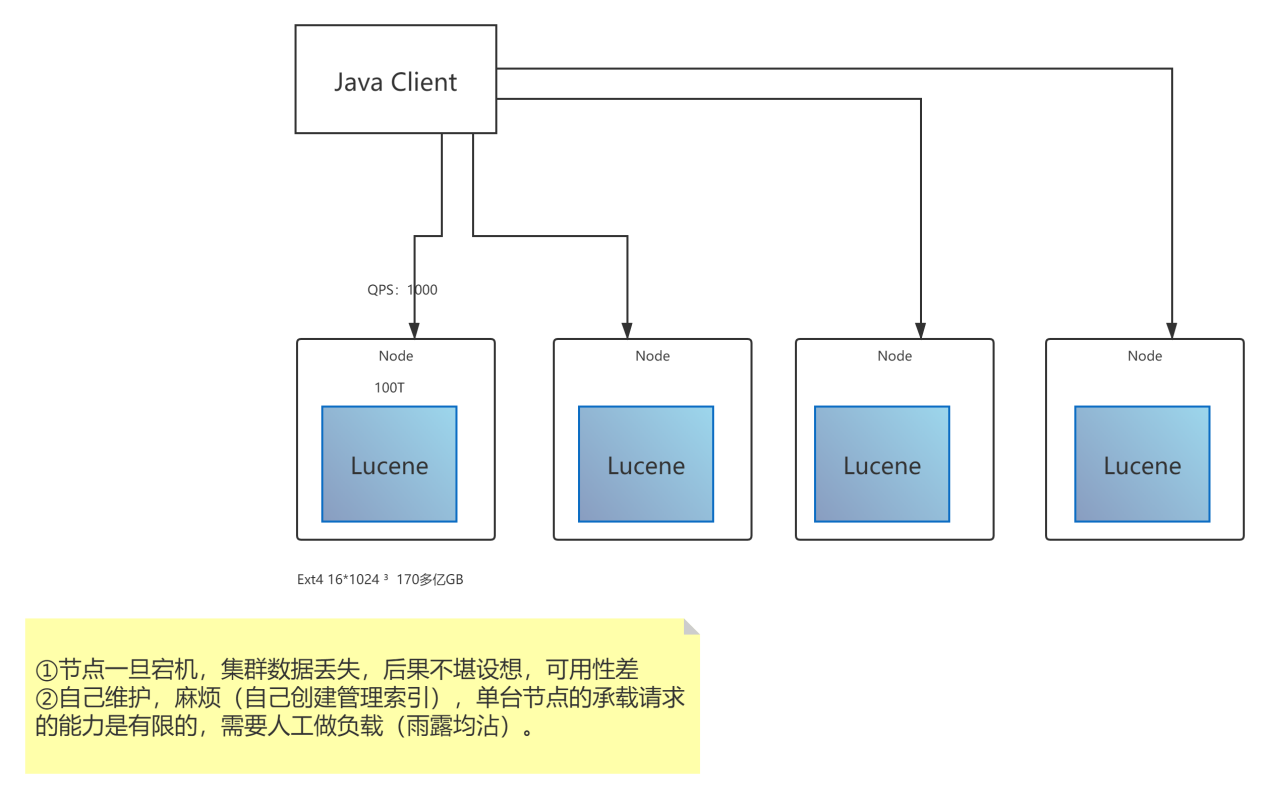
### 3.3 Lucene

#### 3.3.1概念：

Lucene是一个jar包。帮我们创建了倒排索引，提供了复杂的API

#### 3.3.2 Lucene做集群实现搜索存在问题

1. 节点一旦宕机，节点数据丢失，后果不堪设想，可用性差。
2. 自己维护，麻烦（自己创建管理索引），单台节点的承载请求的能力是有限的，需要人工做负载（雨露均沾）。



## Elasticsearch：

### 4.1概念

**分布式、高性能、高可用、可伸缩、易维护** ES≠搜索引擎

### 4.2优点：

1. 面向开发者友好，屏蔽了Lucene的复杂特性，集群自动发现（cluster discovery）
2. 自动维护数据在多个节点上的建立
3. 会帮我做搜索请求的负载均衡
4. 自动维护冗余副本，保证了部分节点宕机的情况下仍然不会有任何数据丢失
5. ES基于Lucene提供了很多高级功能：复合查询、聚合分析、基于地理位置等。
6. 对于大公司，可以构建几百台服务器的大型分布式集群，处理PB级别数据；对于小公司，开箱即用，门槛低上手简单。
7. 相遇传统数据库，提供了全文检索，同义词处理（美丽的cls>漂亮的cls），相关度排名。聚合分析以及海量数据的近实时（NTR）处理，这些传统数据库完全做不到。

### 4.3 应用领域

①百度（全文检索、高亮、搜索推荐）

②各大网站的用户行为日志（用户点击、浏览、收藏、评论）

③BI（Business Intelligence商业智能），数据分析：数据挖掘统计。

④Github：代码托管平台，几千亿行代码

⑤ELK：Elasticsearch（数据存储）、Logstash（日志采集）、Kibana（可视化）

## 5.ES核心概念

1. cluster（集群）：每个集群至少包含两个节点.
2. node：集群中的每个节点，一个节点不代表一台服务器
3. field：一个数据字段，与index和type一起，可以定位一个doc
4. document：ES最小的数据单元 Json

{  
    **"id"**:**"1"**,  
    **"name"**:**"小米"**,  
    **"price"**:{  
        **"标准版"**:**3999**,  
        **"尊享版"**:**4999**,  
        **"吴磊签名定制版"**:**19999**  
    }  
}

1. Type：逻辑上的数据分类，es 7.x中删除了type的概念
2. Index：一类相同或者类似的doc，比如一个员工索引，商品索引。

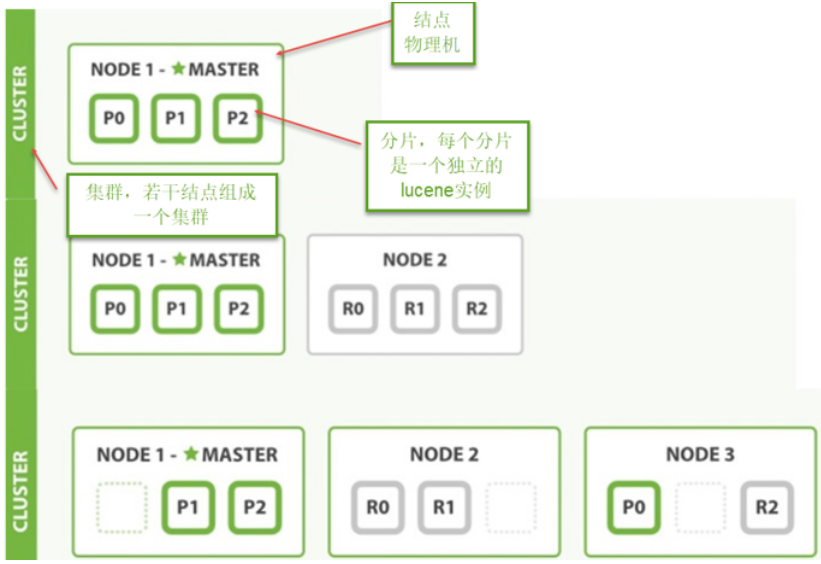
**关系数据库和ES类比**

|  |  |
| --- | --- |
| Elasticsearch | 关系型数据库 |
| Database | Index |
| Tables | Type |
| Row | Document |
| Column | Field |

## 集群管理

### 7.1集群结构

ES通常以集群方式工作，这样做不仅能够提高 ES的搜索能力还可以处理大数据搜索的能力，同时也增加了系统的容错能力及高可用，ES可以实现PB级数据的搜索。



#### 集群概念：

**1、节点：**

ES集群由多个服务器组成，每个服务器即为一个Node节点。

**2、分片：**

当我们的文档量很大时，由于内存和硬盘的限制，同时也为了提高ES的处理能力、容错能力及高可用能力，我们将索引分成若干分片，每个分片可以放在不同的服务器，这样就实现了多个服务器共同对外提供索引及搜索服务。

一个搜索请求过来，会分别从各各分片去查询，最后将查询到的数据合并返回给用户。

**3、副本**

为了提高ES的高可用同时也为了提高搜索的吞吐量，我们将分片复制一份或多份存储在其它的服务器，这样即使当前的服务器挂掉了，拥有副本的服务器照常可以提供服务。

1. **主节点**

一个集群中会有一个或多个主节点，主节点的作用是集群管理，比如增加节点，移除节点等，主节点挂掉后ES会重新选一个主节点。

1. **结点转发**

每个结点都知道其它结点的信息，我们可以对任意一个结点发起请求，接收请求的结点会转发给其它结点查询数据

### shared分片

1：一个index包含多个Shard，默认5P，默认每个P分配一个R，P的数量在创建索引的时候设置，如果想修改，需要重建索引。

2：每个Shard都是一个Lucene实例，有完整的创建索引的处理请求能力。

3：ES会自动在nodes上为我们做shard 均衡。

4：一个doc是不可能同时存在于多个PShard中的，但是可以存在于多个RShard中。

5: P和对应的R不能同时存在于同一个节点，所以最低的可用配置是两个节点，互为主备。